REPORT DOCUMENTATION PAGE	Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY) 20-06-2008	2. REPORT TYPE Final Report	3. DATES COVERED (From - To) 29-May-2007 - 28-Feb-2008

4. TITLE AND SUBTITLE Benchmark Intelligent Agent Systems for Distributed Battle Tracking: Final Report	5a. CONTRACT NUMBER W911NF-07-1-0367
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 622782
6. AUTHORS Albert Esterline	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES AND ADDRESSES North Carolina A&T State University Office of Sponsored Programs North Carolina A&T State University Greensboro, NC 27411 -	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSOR/MONITOR'S ACRONYM(S) ARO
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) 52709-CI-II.1

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
An intelligent multiagent system that maintains distributed situational awareness was developed for the U.S. Army CERDEC C2D (Command and Control Directorate) as part of a distributed battle command for tracking system. The benchmark DBTS (Distributed Battle Tracking System) was implemented using the JADE multiagent framework. Each element that moves as a single entity (such as a tank, foot solider, or helicopter) is endowed with a standard set of agents. The DBTS allows multiple entities to alert each other when they diverge from the common plan and to establish a new common plan. The agents perform JADE-based platform-to-platform communication to communicate with other mission entities. To exploit the ubiquity

15. SUBJECT TERMS
Situation awareness, Multiagent Systems, Rule engine, Web services

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Albert Esterline
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	SAR		19b. TELEPHONE NUMBER 336-334-7245

Report Title

Benchmark Intelligent Agent Systems for Distributed Battle Tracking: Final Report

ABSTRACT

An intelligent multiagent system that maintains distributed situational awareness was developed for the U.S. Army CERDEC C2D (Command and Control Directorate) as part of a distributed battle command for tracking system. The benchmark DBTS (Distributed Battle Tracking System) was implemented using the JADE multiagent framework. Each element that moves as a single entity (such as a tank, foot solider, or helicopter) is endowed with a standard set of agents. The DBTS allows multiple entities to alert each other when they diverge from the common plan and to establish a new common plan. The agents perform JADE-based platform-to-platform communication to communicate with other mission entities. To exploit the ubiquity of Web services in the military and other domains, each entity in the benchmark system exposes a standard set of Web services. Jess (Java Expert Shell System) is a rule engine for the Java platform and is an interpreter for the Jess rule language. It is used here to implement policies that maintain distributed situation awareness, such as when to issue warnings and alerts. Simple motion planning techniques are used to establish paths and speeds for the agents; they can also be used for re-planning after an alert has been raised.

List of papers submitted or published that acknowledge ARO support during this reporting period. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

Number of Papers published in peer-reviewed journals: 0.00

(b) Papers published in non-peer-reviewed journals or in conference proceedings (N/A for none)

Number of Papers published in non peer-reviewed journals: 0.00

(c) Presentations

Monica E. Barnette and Janelle C. Mason, A Prototype Intelligent Multiagent Benchmark for a Distributed Situational Awareness System, a graduate oral presentation given at the 22nd Annual Ronald E. McNair Commemorative Symposium, North Carolina A&T State University, Greensboro, NC, Jan. 27-29, 2008.

Number of Presentations: 1.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts): 0

Peer-Reviewed Conference Proceeding publications (other than abstracts):

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts): 0

(d) Manuscripts

Number of Manuscripts: 0.00

Number of Inventions:

Graduate Students

NAME	PERCENT_SUPPORTED
David Reid	1.00
Janelle Mason	0.20
Monica Barnette	0.20
FTE Equivalent:	**1.40**
Total Number:	**3**

Names of Post Doctorates

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Names of Faculty Supported

NAME	PERCENT_SUPPORTED	National Academy Member
Albert Esterline	0.25	No
FTE Equivalent:	**0.25**	
Total Number:	**1**	

Names of Under Graduate students supported

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Student Metrics

This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period:	0.00
The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:......	0.00
The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:......	0.00
Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale):......	0.00
Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:......	0.00
The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense	0.00
The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields:......	0.00

Names of Personnel receiving masters degrees

NAME	
David Reid	
Janelle Mason	
Monica Barnette	
Total Number:	**3**

Names of personnel receiving PHDs

NAME	
Total Number:	

Names of other research staff

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Sub Contractors (DD882)

Inventions (DD882)

Benchmark Intelligent Agent Systems for Distributed Battle Tracking

Principal Investigator: Albert Esterline
Department of Computer Science
North Carolina A&T State University
Greensboro, NC 27411
esterlin@ncat.edu
(336) 334-7245, ext. 462

Contract No.: W911NF-07-1-0367
Funding Period: 5/29/07-2/28/08
Award Amount: $49,638
ARO Grants Officer's Representative: Dr. Cliff Wang, cliff.wang@us.army.mil
Technical Contact: Israel Mayk, CERDEC C2D, Israel.Mayk@us.Army.Mil

List of Illustrations

Statement of the problem studied

This effort addressed the need on the part of the US Army Communications-Electronics Research, Development, and Engineering Center (CERDEC), Command and Control Directorate (C2D) for a distributed battle tracking system (DBTS) to maintain distributed situation awareness. It is envisioned that each entity in a military operation will have a multi-agent system (MAS) networked together as an integral part of C2D's development of distributed battle command tracking (DBCT) technology. This MAS is specifically a distributed battle tracking system (DBTS), maintaining distributed situation awareness. The Java Agent DEvelopment (JADE) framework is a software framework implemented in Java that simplifies implementation of a MAS through middleware compliant with the Foundation for Intelligent Physical Agents (FIPA) specifications. To evaluate the suitability of JADE for implementing a DBTS, a benchmark JADE-based DBTS implementation was sought. Such an effort should also contributes directly to CERDEC's mission and lead to a better understanding of the potential of a MAS command and control environment.

Summary of the most important results

The initial prototype involved only one simulated entity. Later work introduced multiple entities and enhanced the prototype with Web services and policies enforced by a rule engine. We first describe the single-entity prototype and then address the enhancements.

Single-entity Implementation

The first work completed implemented a prototype Distributed Situational Awareness System (DSAS) for only one entity. The agents on a given entity monitor and predict the entity's deviation from the current plan. This may result in a warning to the entity so that it may adjust its plan accordingly without communicating with the other mission members. In more extreme circumstances, projected deviation may trigger an alert to other members of the unit. The goal is to maintain distributed situation awareness within the unit while minimizing communication since communication ties down bandwidth, consumes battery power, and, most critically, potentially provides a signature to the enemy.

The standard agents within any entity are (see figure 1) the Driver Agent, the GPS Agent, the Monitor Agent (RMN), the Predictor Agent (RPR), and the Planner Agent (RPL). Various messages are sent to the Driver and GPS agents. The RDS is an agent intended to forward messages generated within the entity to the RDS agents of other entities, and it is intended to forward messages sent by all other entities to the appropriate agents in the entity in question. RMN receives (via RDS) alive and alert messages from the other entities as well as their plans, and it sends an alive update message to other entities via RDS to keep them up to date on the status of the entity. Also, RMN sends the entity's actual location to RPL and RPR, which sends a prediction to RPL on request. RPL is responsible for the motion plan for the entity, which is intended to be sent (via RDS) to the other entities. Also, when the actual location or time deviates sufficiently from the planned location or time, RPL sends an alert to the other entities via RDS.

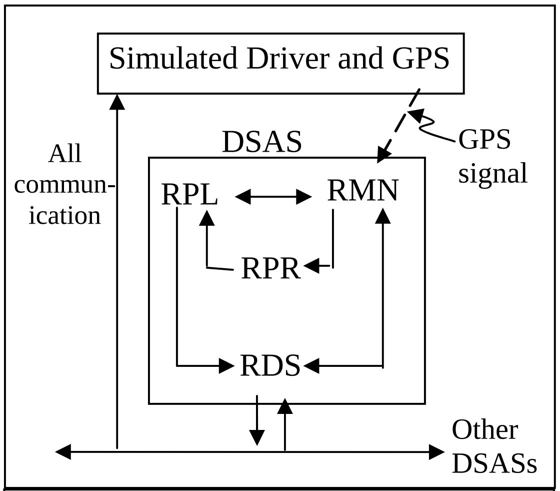

Figure 1. Simulated DSAS

In the actual implementation, RDS is essentially a stub that communicates with the other agents in the DSAS but does not communicate beyond the single-entity DSAS. The plan is provided by a text file read by RPL, and the positions are provided by another text file, which is read by the GPS agent. All agents, however, are implemented as true JADE agents, and all inter-agent communication uses the JADE ACL (Agent Communication Language). Some parameters are adjustable either by the user (namely, distance and time thresholds for Alive messages, distance threshold for Alert messages, and prediction) or by the software engineer through the parameters of the code, such as enabling of debugging or validation testing output and JADE's GUI's for system agents. There are additional optional dummy, sniffer, socket proxy, and log manager agents with GUI's, which can be used to inspect agent interactions and log data on a per package basis.

Validation testing was performed using a *plan file* of waypoints simulating the route the simulated entity is expected to follow and one of twelve files with waypoints simulating GPS signals received for the route taken by the entity. The latter are called *actual-waypoint files*, where "actual" here is in contrast to "predicted." Every test used the same plan file and one of the twelve actual-waypoint files. Each line of a file is a waypoint represented by three integers, the first two providing grid coordinates, and the third being the time coordinate. No units are needed for this simulation.

Each actual waypoint file contains eight waypoints chosen to trigger Alive and Alert messages in a certain pattern. In these test runs, an Alive message was triggered when the entity had moved at least distance 5 from where it was when it sent the last Alive message or the time since the last Alive message was at least 2. An Alert message was sent when the entity had exceeded a distance threshold from the plan segment a certain threshold number of times in a row; for these tests, the distance threshold was 1 and the threshold for the number of consecutive off-plan actual waypoints was 2. The waypoints in the actual-waypoint files were also chosen so that each file contains a transition from one plan segment to another, which requires an actual waypoint within the threshold distance of the plan waypoint where the segment transition occurs. The first six actual-waypoint files were designed to contain actual waypoints that generate Alive and Alert messages in interesting patterns when the current actual position is compared with the planned position for the current time.

The remaining six actual-waypoint files were designed to contain actual waypoints that give rise to predicted waypoints generating interesting patterns of Alive and Alert messages. Prediction is a user selectable feature that uses linear extrapolation to predict the entity's future location. This has the effect of sending Alert messages earlier than they would be sent without prediction if the entity is currently heading in a direction that would take it out of the on-plan region. It also has the corresponding effect of ending Alert messages earlier if the entity is off-plan but heading in a direction that will put it back on-plan.

The DSAS was run on the first six actual-waypoint files with prediction off and then was run again on the same files with prediction on. The DSAS was then run on actual-waypoint files 7-12 only with prediction on since these files were designed so that the predicted points would generate the desired behavior. Note that there was a total of eighteen test runs. Also note that the sequence of Alive messages for a given actual-waypoint file will be the same whether or not prediction is turned on since the RMN agent determines whether an Alive message should be generated before it calculates a prediction (if in fact it does so). The MAS successfully performed all functions specified by the design, and validation testing confirmed that input data was processed and communicated accurately among agents with and without the prediction feature enabled. Transitions from one plan segment to another also functioned as designed.

In summary, each agent performs the role for which it was intended, and all agents worked together to form a modular, easily understood and extensible DSAS that can be run and developed further on any platform capable of running Java. By implementing the agents with the `dummyAgent` class, each agent can be controlled through its GUI to simplify testing and debugging without continually restarting the JADE environment. The GUIs of the `dummyAgent` and other JADE system agents will also be beneficial when testing multiple DSAS as a distributed MAS.

Enhancements with Web Services

To allow entities to expose and to consume Web services, this project is uses the Apache Axis (Apache eXtensible Interaction System) open source SOAP engine. It is a framework for constructing Web service providers and consumers. It supports WSDL, allowing the user easily to build stubs, to access remote services, and to automatically export machine-readable descriptions of deployed services. Axis runs as a servlet to process the incoming message, extract information from the message headers and payloads, and to provide the RPC interface. Apache Tomcat is a popular servlet container and is used in this project. UDDI (Universal Description, Discovery and Integration) is a OASIS yellow-pages standard that defines a way to publish and discover information about Web services. When a service is registered with UDDI, two SOAP messages are exchanged. The first establishes authentication, and the second registers the Web service, whereby the URL and WSDL document are added to the description field. The registry can then be used by service requesters to locate the services they need. The current work uses jUDDI, which is an open source Java-based implementation of a UDDI registry. jUDDI-enabled applications can look up services in the UDDI registry and then proceed to "call" those Web services directly. Also, WSDL documents and URLs are stored persistently in a MySQL database and provided in the registry.

This research is breaking new ground in showing how Web service orchestration can recommend the sequence and coordination of activities, tasks and events for the entities. Shadow agents, which provide cache summaries of information available from corresponding agents on other entities, are introduced to allow entities to track other entities. Finally, a unique system of Web services, agents, and rules maintains distributed situation awareness even under adverse conditions.

Enhancements Involving the Rule Engine

Jess (Java Expert System Shell) is a rule engine and scripting language that supports rule-based as well as procedural programming. This rule engine is dynamic and implemented in Java while providing rule-based reasoning able to provide intelligence for an agent. A major advantage of Jess is its speed and efficiency since it uses the Rete algorithm. Due to its declarative paradigm, Jess can continuously apply a collection of rules to a collection of facts by pattern matching. Jess requires a working memory, a fact base that contains all data that the rule system is working with. It may hold both the premises and the conclusions of the rules. It is up to the designer to determine what should be stored in the working memory. In order to make searching the working memory a very fast operation, one or more indices are maintained on the working memory. Jess maintains an agenda containing the rules that currently may fire. Once the rule engine determines which rule to fire, it executes that rule's action part.

Motion planning is kept simple and is implemented only to provide interesting scenarios for policies. We use techniques based on visibility graphs and convex hulls. For simplicity, we assume that all obstacles are convex polygons. Since there are multiple participants in a mission, the plan for one entity must take into consideration the plans for the other entities, but, aside from the entities participating in the mission, the environment is assumed static. With a collection of entities, we form the convex hull of the vertices of the obstacles. The paths of two of the entities follow opposite sides of this convex hull; the paths of other entities follow edges of the visibility graph that will generally be in the interior of the convex hull.

For re-planning, the visibility graph and convex hull are retained. The extent of re-planning is governed by policies encoded in Jess rules. We can identify two extremes. If the deviation is relatively slight, it might suffice simply to have the divergent entity return to the original path in the shortest time; this might require other entities to slow down for some time. If the deviation is relatively large, or if more than one entity deviates, it might be best to come up with entirely new plans. Note that, in terms of Web services, the coordination suggested here is similar to choreography; this is especially so when expectations of others comes into play.

Policies are encoded as Jess rules and are enforced by the Jess rule engine. The following are several areas for which policies are needed.

When to Send an Alert

The single-entity implementation sends alerts when it is predicted that the distance between the entity and its intended path exceeds a given distance threshold or when the time the entity is at a given point is outside a given time threshold. There aree other conditions under which an alert could be essential for maintaining distributed situation awareness, such as when an entity discovers that a bridge that several entities intended to use is impassable or when the unanticipated presence of the enemy is detected. In general, alerts should not be suppressed in the interest of reducing an entity's signature since they generally signal critical conditions.

Something else that is useful is to have an alert issued by one entity trigger an alert by another entity. For example, if one entity sends an alert (along with its anticipated location) because it is about to exceed its distance threshold, an entity receiving this alert might determine that there is no way the first entity could get back to the original path and maintain a relation to it without itself modifying its plan. Cascading alerts like this could indicate the need for more drastic revision of the plan, perhaps with help from mission-level resources.

When to Ask for Information and What to Ask For

Entities should consume services sparingly since the communication would increase their signatures. Still, a certain amount of information must be shared for the entities to maintain a common plan. During re-planning, entities must share their plans and priorities, and certain central services must be used. These central services include orchestration (for global coordination, for example, to ensure that all regions are searched) and global alerting (for example, to define no-go areas). Policies must be developed for which agents provide which services.

What to do with Alerts from Others

When an alert is received from another entity, comprehending its significance is a question of Level 2 situation awareness (SA). To achieve this, there must be rules that establish what actions to perform and how to update working memory. For distributed SA, these rules will capture expectations about the behavior of other entities.

This research is breaking new ground in formulating and enforcing rules within a multiagent system to maintain distributed situation awareness in a group of entities conducting a military mission. The rules establish policies for raising and responding to alerts and for consuming Web services. Particularly significant are the policies for various extents of re-planning in response to alerts, although the techniques for planning and re-planning are kept simple. The policies enforced in the system amount to a kind of choreography from the point of view of the Web services.